Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI-2693: Laboratorio de Algoritmos III
Trimestre Septiembre-Diciembre 2018

Proyecto III
USB-DataFlow

1. Introducción

En 1979, Dan Bricklin y Bob Frankston escribieron VisiCalc, la primera aplicación de hojas de cálculo. Dicha aplicación se convirtió en un gran éxito y, en esa época, fue la mejor aplicación para los computadores Apple II. Hoy en día, las hojas de cálculo se encuentran en la mayoría de los computadores de escritorio en programas como excel, google spreadsheets, libreoffice, entre otros.

La idea detrás de las hojas de cálculo es muy sencilla, pero poderosa. Una hoja de cálculo consiste de una tabla donde cada celda contiene un número o una fórmula. Una fórmula puede computar una expresión que depende de los valores de otras celdas. Texto y gráficos pueden ser añadidos para propósitos presentacionales.

El objetivo de este proyecto es que usted escriba USB-DataFlow. Un programa que simule el funcionamiento de las hojas de cálculo que solo sirve para calcular valores i.e., no tiene formas de añadir títulos, gráficos, entre otros. El proyecto se va a dividir en dos partes para facilitar la elaboración del mismo.

2. Detalles de USB-DataFlow

2.1. Detalle de la hoja de cálculo

En la mayoría de los programas actuales de hojas de cálculo la misma es una matriz bidimensional de filas y columnas *infinitas*. Para este proyecto se le va a especificar el tamaño en filas y columnas de la tabla mediante un

	A	B	C	D	E
1	1	20	6	1	$= min(A1, D1)$
2		$= A1 * max(0, E1)$	$= SUM(B1)$		
3		-1		$= MIN(b1, B2)$	
4	0				$= B2 + B2$
5	$= 20 * 59 + 0$	$= E1 + B2$	5		

Cuadro 1: Hoja de Cálculo de Ejemplo

archivo (Ver sección 3.2.1). En la tabla 1 puede ver un ejemplo de una hoja de cálculo.

Cada celda esta compuesta de una *fórmula* (sección 2.2). Si la celda está vacía, se supone que su valor es 0.

2.2. Detalles de las fórmulas

En esta sección se describe a detalle como armar las fórmulas que puede recibir cada celda de las hojas de cálculo de USBDataFlow

2.2.1. Operador

Un *Operador* es un operador matemático regular. Las operaciones matemáticas posibles son:

- Multiplicación ($*$)
- Adición ($+$)
- Substracción ($-$)

Las precedencias entre los mismos son las mismas que en la matemática regular.

2.2.2. Valor

Un *Valor* es el componente más atómico que puede contener una celda. El valor de esta va a ser el valor asociado al *Valor*. Los posibles valores son:

- Funciones integradas (ver sección 2.3).
- Literales enteros.
- Etiquetas a celdas.

Ejemplos:

- 1
- $MIN(A1, A2)$
- $A1$

2.2.3. Expresión

Una *Expresion* es lo que nos permite componer valores usando los operadores binarios. Una *Expresion* puede ser:

- Un *Valor* ó
- Una *Expresion* seguido de un *Operador* seguido de un *Valor*

Nótese que la definición de *Expresión* es recursiva. Ejemplos:

- $1 * 2 - 3$
- $A1$
- $C1 * 3$
- $MAX(A1, -1)$

2.2.4. Fórmula

Una *Fórmula* es una *Expresión* precedida por un símbolo igual (=).

2.3. Funciones integradas

USB-DataFlow trae en sus hojas de cálculo 4 funciones integradas (built-in). Dichas funciones son muy básicas

2.3.1. `MIN`

La función `MIN` toma 2 números enteros como argumentos y retorna el menor entre los dos

2.3.2. `MAX`

La función `MAX` es idéntica a la función MIN (sección 2.3.1) a diferencia que retorna el mayor entre los dos argumentos

2.3.3. `SUM`

La función `SUM` recibe como parámetro un entero n y retorna la suma de los números enteros desde 1 hasta n si este es positivo. Si n es negativo, se retorna la suma desde -1 hasta n.

```
print(sum(4))
// 10
print(sum(-5))
// -15
```

3. Actividades a realizar

3.1. Primera Parte: Evaluador

La primera parte del proyecto consiste en construir un evaluador de expresiones aritméticas. Un evaluador toma como input una *Expresion* y retorna como output el valor correspondiente a la misma. Por ejemplo:

- $1 * 2 - 3$ evalúa a -1
- $1 - 2 * 3$ evalúa a -5
- $1 + 1 + 1 + 1 * 1 - 1$ evalúa a 3
- $20 - 2 * 20 + 10$ evalúa a -10
- $MAX(SUM(3), SUM(5))$ evalúa a 15

3.1.1. Entrada de Datos

El programa se debe poder ejecutar desde la consola ejecutando el siguiente comando:

```
>java Evaluador <archivodeexpresiones>
```

Donde *<archivodeexpresiones>* es un archivo que contiene múltiples expresiones, una por línea. En la expresión no hay espacios intermedios. Es importante recalcar que los valores de las expresiones no van a tener referencias a celdas, sólo números. Es decir, $A1+2$ no lo puede manejar el evaluador (en esta primera parte) pues no sabe el valor a priori de $A1$. El formato del mismo sería el siguiente:

```
expresion1
expresion2
expresion3
...
expresionn
```

3.1.2. Salida de Datos

El programa debe imprimir por cada expresión en el archivo de input el valor de la misma, una en cada línea, en el mismo orden en el que vino el input.

3.1.3. Ejemplo

Dado el siguiente archivo de Entrada

```
1*2-3
1-2*3
1+1+1+1*1-1
20-2*20+10
MAX(SUM(3),SUM(5))
```

Se obtiene el siguiente resultado al correr la aplicación

```
>java Evaluador expresiones.txt
-1
-5
3
-10
15
```

3.2. Segunda Parte: USB-DataFlow

La segunda (y última) parte del proyecto consiste en la aplicación de hojas de cálculo. Para evaluar las expresiones de cada celda debe usar la primera parte.

3.2.1. Entrada de Datos

El programa se debe poder ejecutar desde la consola ejecutando el siguiente comando:

```
>java USBDataFlow <hojadecalculo>
```

Donde *<hojadecalculo>* es un archivo que contiene la configuración dada por el usuario de la hoja de cálculo. Dicha configuración puede ser errónea y contener ciclos e.g., el valor la celda A1 depende del valor de la celda A2 **y** viceversa.

El formato del archivo de entrada es como sigue:

```
n m
celda11 celda12 ... celda1m
celda21 celda22 ... celda2m
...      ...    ... ...
celdan1 celdan2 ... celdanm
```

- En la primera línea dos enteros n y m
- En cada una de las siguientes n líneas se imprimen m expresiones, Indicando lo que escribió el usuario en la hoja de cálculo.

Las celdas son representadas por *letranumero*, donde la letra (en mayúscula o minúscula) representa la columna y el número la fila. Solo letras del alfabeto inglés son permitidas (es decir la ñ no puede aparecer). La enumeración de las letras es la siguiente: A, B, C, . . . , Z, AA, AB, AC, . . . , AZ, BA, . . . , BZ, CA, . . . , ZZ, AAA, AAB, . . . , AAZ, ABA, . . . , ABZ, ACA, . . . , ZZZ.

3.2.2. Salida de Datos

El programa debe procesar la hoja de cálculo suministrada e imprimir en la consola los cálculos resultantes. El programa debe ser capaz de detectar ciclos, indicandole al usuario que la configuración es errónea y no mostrar ningún cálculo.

En caso que la hoja de cálculo suministrada no contenga ciclos, se debe imprimir la hoja con los valores finales

valor(celda$_{11}$) valor(celda$_{12}$) ... valor(celda$_{1m}$)
valor(celda$_{21}$) valor(celda$_{22}$) ... valor(celda$_{2m}$)
... ...
valor(celda$_{n1}$) valor(celda$_{n2}$) ... valor(celda$_{nm}$)

Si hay ciclos, se le debe notificar al usuario que hay un ciclo y cuales son los vertices en el mismo:

```
Su configuración contiene un ciclo:
(imprimir celdas separadas por " -> ")
```

Si hay varios ciclos, imprima cualquiera de ellos.

3.2.3. Ejemplo

Caso sin ciclos Dado el siguiente archivo de Entrada

```
3 3
1 2 3
=A1+A2+a3 =b1+B1 0
0 0 =MiN(B2, MAX(B3, B1))
```

Se obtiene el siguiente resultado al correr la aplicación

```
>java USBDataFlow hojadecalculo.txt
1 2 3
6 12 0
0 0 6
```

Caso con ciclos Dado el siguiente archivo de Entrada

```
2 2
=A1 2
2 1
```

Se obtiene el siguiente resultado al correr la aplicación

```
>java USBDataFlow hojadecalculoconciclos.txt
Su configuración contiene un ciclo:
A1 -> A1
```

4. Requerimientos de su Entrega

Usted debe de hacer uso de las interfaces de Grafos (Dirigido o no Dirigido dependiendo de lo que necesite) de su proyecto 1. Debe hacer la implementación de dichas interfaces con la estructura de datos más adecuada que considere.

Sus implementaciones deben ser razonablemente eficientes. Todo el código debe estar debidamente documentado. Se deben indicar una descripción de los métodos, la descripción de los parámetros de entrada y salida y el orden

de ejecución de cada método aplicando el estándar para la documentación de código en JAVA. Su implementación debe incluir manejo de excepciones. Puede usar las librerías de JAVA que considere útiles. Su código debe hacer uso de la guía de estilo publicada en el Aula Virtual.

5. Entrega

El proyecto debe ser subido al Moodle de la materia en la sección marcada como "Proyecto 3" hasta el 6 de diciembre de 2018. Deberá entregar la Declaración de autenticidad de entregas. Sólo deberá efectuar una entrega por grupo.

6. Evaluación

El proyecto tiene una ponderación de 20 puntos. El programa debe correr sin errores. Se asignarán:

- 10 puntos por Código
  - 3 puntos por Construir correctamente el grafo de cada expresión
  - 3 puntos por Construir el grafo de dependencia entre celdas
  - 4 puntos por Algoritmo de orden topológico / arbol mínimo cobertor
- 7 puntos por Ejecución
  - 2 puntos por Evaluar correctamente las expresiones según el orden de operaciones
  - 4 puntos por Evaluar correctamente expresiones con Sum, Min y Max
  - 1 puntos por Imprimir en forma de tabla
- 3 puntos por Documentación
  - 1 puntos por cómo construye grafo expresión
  - 1 puntos por cómo construye grafo dependencia
  - 1 puntos por Algoritmo